Đề xuất cấu trúc cây phát hiện xung đột trong tập luật của tường lửa

Nguyễn Mạnh Hùng[1], Vũ Duy Nhất[2]
[1]Phòng Sau đại học, Học viện Kỹ thuật Quân sự
[2]Cục Cơ yếu, Bộ Tổng tham mưu

Tác giả liên hệ: Vũ Duy Nhất, nhatbest@gmail.com
Ngày nhận bài: 31/05/2017, ngày sửa chữa: 11/04/2018, ngày duyệt đăng: 25/12/2018
Xem sớm trực tuyến: 28/12/2018, định danh DOI: 10.32913/rd-ict.vol3.no40.478
Biên tập lĩnh vực điều phối phản biện và quyết định nhận đăng: PGS. TS. Nguyễn Khánh Văn

Tóm tắt: Tường lửa là một thiết bị bảo mật mạng, trong đó sử dụng tập luật để kiểm soát các gói tin đi qua thiết bị. Cấu hình các luật tường lửa là nhiệm vụ rất khó khăn ngay cả đối với các chuyên gia bảo mật, đặc biệt đối với các hệ thống mạng phức tạp. Sai sót trong quá trình cấu hình thiết bị sẽ tác động tới hai khía cạnh: (i) làm ảnh hưởng tới sự an toàn của hệ thống mạng cần được bảo vệ và (ii) làm suy giảm năng lực xử lý của thiết bị tường lửa. Bài báo này đề xuất cấu trúc cây phát hiện xung đột (CDT: Conflict Detection Tree) có khả năng phát hiện tất cả các loại xung đột trong một tập luật của tường lửa một cách hiệu quả. Tính chính xác và tính hiệu quả của cấu trúc CDT được giới thiệu và chứng minh chi tiết trong bài báo. Cấu trúc CDT được triển khai và kiểm chứng với dữ liệu thực tế, cho thấy tính khả dụng của nó.

Từ khóa: *Tường lửa, an ninh mạng, tiền tố, xung đột, chính sách an ninh.*

Title: **A New Conflict Detection Tree Structure in the Firewall Rule Set**

Abstract: Firewall is a network security device that uses rules to control incoming and outgoing network traffic. Configuring firewall rules is a very difficult task even for network security experts, especially for complex networks. Mistakes made in the configuration process will cause two damaging effects: (i) affecting the security of the network that needs protection, and (ii) reducing the performance of the firewall device. This article will introduce a Conflict Detection Tree (CDT) structure that effectively detects all conflicts in a firewall rule set. The accuracy and effectiveness of the CDT structure is presented and substantiated in the article. The proposed CDT structure has been implemented and tested with real data.

Keywords: *Firewall, network security, firewall rules, conflict, security policy.*

I. GIỚI THIỆU

Tường lửa là một trong các loại thiết bị không thể thiếu trong việc bảo đảm an ninh và an toàn cho các hệ thống mạng. Thiết bị này có chức năng ngăn cách và bảo vệ cho một mạng nội bộ của một đơn vị hay một tổ chức với các mạng công cộng hay mạng của các tổ chức khác. Các gói tin khi đi qua tường lửa được kiểm soát theo cả chiều vào và chiều ra. Mỗi thiết bị tường lửa được trang bị một chính sách an ninh cho mục đích kiểm soát các gói tin nêu trên. Chính sách an ninh này tồn tại trên thiết bị tường lửa dưới dạng một tập luật do người quản trị thiết lập. Mỗi luật trong tập luật gồm các giá trị điều kiện của các trường thông tin trong header của gói tin cần thỏa mãn, và một trường quan trọng là hành động của luật đó đối với gói tin thỏa mãn. Hành động này có thể là một trong hai giá trị: Accept (cho phép gói tin đi qua) hoặc Deny (không cho phép gói tin đi qua).

Một tập luật có thể được xây dựng bởi một người quản trị khi triển khai thiết bị và nó có thể được bổ sung hay xóa bỏ các luật ngay khi có sự thay đổi về chính sách an ninh trong quá trình vận hành hệ thống. Số lượng các luật trong tập luật tỷ lệ thuận với độ phức tạp của chính sách an ninh được triển khai trên thiết bị. Trong thực tế hiện nay, với việc phát triển mạnh mẽ về quy mô hệ thống và về số lượng các loại hình dịch vụ triển khai, chính sách an ninh cần được triển khai trên các thiết bị tường lửa ngày càng phức tạp. Điều này cũng đồng nghĩa với việc tập luật trong chính sách an ninh mà người quản trị phải triển khai ngày càng tăng lên về số lượng luật và phức tạp về cấu trúc. Nhiệm vụ xây dựng và quản lý chính sách an ninh cho tường lửa trở nên khó khăn hơn.

Các luật trong tập luật có thể xung đột lẫn nhau như dư thừa, mâu thuẫn về hành động,... Các xung đột trong tập luật có thể làm ảnh hưởng trực tiếp đến an ninh của

hệ thống khi cho phép các gói tin không hợp pháp đi qua, hoặc ảnh hưởng đến hoạt động bình thường của hệ thống mạng khi loại bỏ các gói tin hợp lệ, hay sẽ làm ảnh hưởng đến hiệu năng (về mặt lưu trữ và xử lý) của chính thiết bị tường lửa khi tồn tại các luật dư thừa.

Tại thời điểm năm 2004, kết quả khảo sát được trình bày trong [1] cho thấy số lượng lớn các xung đột trong các tập luật của tường lửa là một thực tế phải chấp nhận, và hiện nay con số này chắc chắn sẽ lớn hơn rất nhiều. Chính vì vậy, việc nghiên cứu để phát hiện và xử lý các xung đột trong tập luật của tường lửa là một vấn đề đã và đang được nhiều nhà nghiên cứu quan tâm thực hiện.

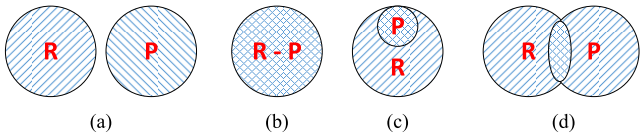
Các kỹ thuật phát hiện xung đột trong tập luật của các thiết bị mạng nói chung và tường lửa nói riêng có thể được chia làm hai loại cơ bản khi căn cứ vào số chiều của tập luật đó: (i) phát hiện xung đột trên các tập luật hai chiều và (ii) phát hiện xung đột trên các tập luật nhiều chiều.

Các kỹ thuật phát hiện xung đột trên các tập luật hai chiều tiêu biểu gồm có [2–9]. Các kỹ thuật này chỉ thực hiện kiểm tra, phát hiện và xử lý các xung đột trong tập luật với hai chiều địa chỉ IP nguồn và địa chỉ IP đích trên các thiết bị mạng nói chung như thiết bị định tuyến, tường lửa, bảo mật IPSec, v.v. Các kỹ thuật này có hạn chế là không thể áp dụng cho trường hợp các tập luật có số chiều lớn hơn và kiểu dữ liệu của các chiều bị hạn chế.

Các kỹ thuật dạng thứ hai bao gồm các công trình [10–14]. Các kỹ thuật này đưa ra các giải pháp nhằm phát hiện và xử lý các xung đột giữa các luật nhiều chiều trong tập luật của thiết bị tường lửa. Riêng nhóm các tác giả trong công trình [10] đề xuất hướng phát hiện và xử lý các xung đột giữa các luật trên tập luật của một nhóm các thiết bị tường lửa nằm trên đường di chuyển của các gói tin. Các kỹ thuật này được đánh giá mang tính tổng quát hơn các kỹ thuật dạng đầu.

Bài báo này đề xuất cấu trúc cây phát hiện xung đột (CDT: Conflict Detection Tree) nhằm phát hiện các xung đột trong tập luật theo nhiều chiều trên một thiết bị tường lửa đơn lẻ. Cấu trúc CDT tối ưu về mặt lưu trữ, có thể thực hiện với các dạng dữ liệu khác nhau của mỗi trường, có ưu điểm về mặt thời gian trong quá trình xây dựng cây cũng như phát hiện xung đột. Ưu điểm của cấu trúc CDT so với các cấu trúc khác được chứng minh bằng lý thuyết và đánh giá qua quá trình thực nghiệm.

Bài báo được tổ chức với các phần tiếp theo như sau. Mục II giới thiệu kiến thức chung về xung đột và một số thuật toán phát hiện và xử lý xung đột trong tập luật nhiều chiều. Mục III đề xuất cấu trúc CDT cho việc phát hiện các xung đột. Mục IV trình bày kết quả thử nghiệm và đánh giá thuật toán đề xuất. Mục V là kết luận và xác định các hướng nghiên cứu có thể tiếp tục.



Hình 1. Các mối quan hệ hình học của không gian luật của hai luật: (a) R và P tách biệt nhau hoàn toàn; (b) R trùng khớp hoàn toàn với P; (c) R chứa P; (d) R và P có một phần giao nhau.

II. CÁC KIẾN THỨC LIÊN QUAN

1. Một số khái niệm

1) Định dạng luật trong tường lửa:

Mỗi luật trong tập luật của tường lửa được tạo bởi tập giá trị mẫu của các trường và một hành động đi kèm. Tập giá trị mẫu này chính là điều kiện các trường thông tin trong gói tin cần thỏa mãn nếu muốn khớp với luật này. Các trường thông tin ở đây có thể là bất cứ trường nào của IP, UDP hay TCP headers. Trong thực tế các trường này thường là IP nguồn, IP đích, cổng nguồn, cổng đích và kiểu giao thức. Hành động gắn mỗi luật có thể là Accept (cho phép gói tin đi qua) hoặc Deny (cấm đi qua).

Về mặt hình thức, mỗi luật R có thể được biểu diễn bởi R($f_1, f_2, \ldots, f_n, Action$), trong đó f_i là giá trị mẫu của trường thứ i. Giá trị mẫu có thể được cho dưới dạng khoảng, tiền tố, hay tập các giá trị khác nhau.

2) Không gian luật:

Không kể đến trường *Action* trong luật, xét trong không gian n chiều thì mỗi luật sẽ thuộc một điểm hay một đa diện hình học trong không gian đó. Mối quan hệ giữa hai không gian của luật R và luật P sẽ thuộc một trong 4 trường hợp được thể hiện ở hình 1.

3) Các loại xung đột trong tập luật tường lửa:

Các loại xung đột giữa các luật của tường lửa được xác định từ mối quan hệ giữa không gian luật, thứ tự và hành động của chúng. Al-Shaer và các cộng sự trong [10] đã định nghĩa và công thức hóa năm loại mối quan hệ giữa hai không gian luật, bao gồm phân tách hoàn toàn (Completely Disjoint), khớp hoàn toàn (Exactly Matched), bao gồm (Inclusively Matched), phân tách vài phần (Partially Disjoint) và tương quan (Correlated).

Tuy nhiên các tác giả trong công trình [12] đã chỉ ra rằng có thể coi quan hệ phân tách hoàn toàn và phân tách vài phần là một. Bốn mối quan hệ còn lại tương ứng với bốn trường hợp mô tả hình học trong hình 1. Từ các mối quan hệ trên, các tác giả trong các công trình [10] và [12] đã chỉ ra bốn loại xung đột có thể có giữa hai luật, đó là kiểu bóng (Shadowing), kiểu tương quan (Correlation), kiểu tổng quát (Generalization), kiểu dư thừa (Redundancy).

Trong các kiểu xung đột trên, chúng ta có thể bỏ qua kiểu tổng quát [12, 13].

Kiểu bóng: Luật R1 là bóng của luật hay một nhóm luật R2 trước nó khi tất cả các gói tin khớp với R1 cũng khớp với R2 và hành động của R1 khác với hành động của R2. Mối quan hệ không gian luật giữa R1 và R2 tương ứng với kiểu xung đột này là R2 trùng khớp với R1 hoặc R2 chứa R1. Trong trường hợp này, tất cả các gói tin mà một luật muốn cấm có thể lại được cho phép bởi các luật trước nó. Do đó, luật bị bóng sẽ không bao giờ có tác dụng.

Kiểu tương quan: Luật R1 tương quan với luật R2 nếu một phần gói tin khớp với R1 cũng khớp với R2 và hành động của R1 và R2 là khác nhau. Mối quan hệ không gian luật giữa R1 và R2 tương ứng với kiểu xung đột này là R2 và R1 có một phần giao nhau. Trong trường hợp này, các gói tin khớp bởi phần chung giữa hai luật này có thể được cho phép bởi luật này nhưng lại bị cấm bởi luật khác.

Kiểu dư thừa: Luật R1 là dư thừa khi tồn tại một luật hay một nhóm luật R2 trước nó thỏa mãn điều kiện tất cả các gói tin khớp với R1 cũng khớp với R2 và hành động của R1 và R2 là giống nhau. Mối quan hệ không gian luật giữa R1 và R2 tương ứng với kiểu xung đột này là R2 trùng khớp với R1 hoặc R2 chứa R1. Kiểu xung đột này không ảnh hưởng đến an ninh của thiết bị nhưng làm lãng phí không gian lưu trữ các luật.

2. Một số kỹ thuật phát hiện xung đột trong tập luật của tường lửa

1) Kỹ thuật FIREMAN:

Kỹ thuật FIREMAN được các tác giả trong công trình [11] đề xuất nhằm phát hiện xung đột giữa các luật trong tập luật trên một tường lửa đơn hay các tường lửa trong một phân đoạn mạng. Trong FIREMAN, các luật được lưu trữ bởi biểu đồ quyết định nhị phân (BDDs: Binary Decision Diagrams). Kỹ thuật này phát hiện các xung đột trong tập luật bằng cách phân tích mối quan hệ giữa một luật cụ thể với các tập hợp các khoảng giá trị của gói tin phù hợp với các luật đứng trước nó. Điều này làm cho FIREMAN có hạn chế là với mỗi luật nó chỉ kiểm tra các luật trước đó mà bỏ qua tất cả các luật đứng sau khi thực hiện phân tích xung đột.

2) Kỹ thuật phân mảnh:

Kỹ thuật phân mảnh (Rule-Based Segmentation) được các tác giả trong công trình [13] đề xuất. Trong đó, kỹ thuật đi sâu vào phát hiện và xử lý các xung đột giữa các luật qua việc tìm kiếm phần không gian luật giao nhau của các luật. Các tác đã đưa ra thuật toán phân tách không gian luật của tất cả các luật thuộc tập luật thành các miền tách biệt trong đó gồm các miền không giao nhau và các miền giao nhau. Trong các miền giao nhau, kỹ thuật phân mảnh chia thành ba loại không gian, đó là Allow (cho các luật có hành động là Allow), Deny (cho các luật có hành động là Deny) và Conflict (các miền không gian của các luật mà có hành động khác nhau) và xác định chỉ có miền Conflict là chứa các luật xung đột. Với cách phân chia như vậy, kỹ thuật phân mảnh có hạn chế là sẽ bỏ qua các loại xung đột kiểu dư thừa.

Một hạn chế nữa của kỹ thuật phân mảnh trong công trình [13] là đưa ra thuật toán phân mảnh không gian luật của các luật đầu vào nhưng vẫn để cốt lõi là việc tính toán xác định các biên của mỗi mảnh không gian luật đó theo các chiều cụ thể của mỗi luật không được các tác giả mô tả chi tiết, điều này làm giảm tính thuyết phục của đề xuất.

3) Cấu trúc FAT:

Các tác giả của công trình [14] đã đề xuất xây dựng cấu trúc cây mới có tên là FAT (Firewall Anormaly Tree), nhằm phát hiện và xử lý các xung đột trong tập luật của tường lửa. Trong công trình [14], mỗi trường của một luật được phân tách thành các *element* và mỗi *element* lưu trữ thông tin về một byte của trường có định dạng là $((byte, mask)_b, (dim, ord)_o)$. Trong đó, *mask* là số bít được sử dụng trong byte đang xét, *byte* là giá trị của *mask* bit của byte, *dim* là trường đang xét (1: source IP, 2: destination IP, v.v.), *ord* là vị trí của byte đang xét trong trường *dim*. Các tác giả đưa ra định nghĩa về thứ tự giữa các *element* với nhau, trong đó *element* đứng trước khi có *mask* lớn hơn, trong trường hợp giá trị này bằng nhau thì *element* nào có *dim* nhỏ hơn sẽ đứng trước.

Cấu trúc FAT được xây dựng từ tập tất cả các *element* của các luật. Một luật được biểu diễn trên cấu trúc FAT bằng một đường dẫn luật gồm các nút, mỗi nút được xây dựng dựa trên thông tin của các *element*. Các *element* đứng trước sẽ được ưu tiên lựa chọn trước cho việc xây dựng các nút. Mỗi nút sẽ có ba tập, đó là tập **P** (Primary) chứa các luật khớp với đường dẫn từ nút gốc đến nút đang xét, tập **S** (Second) chứa các luật có không gian luật chứa không gian không gian luật ở tập **P** và tập **T** (Tertiary) chứa các luật có không gian luật giao với không gian không gian luật ở tập **P**. Việc xây dựng cấu trúc FAT sẽ được thực hiện đến khi tất cả các *element* đã được chọn hết và mối quan hệ giữa các luật được xác định nhờ các tập **P**, **S** và **T** tại các nút lá.

Kỹ thuật này có các hạn chế sau đây:

- Phức tạp khi áp dụng cho các trường dữ liệu được cho dưới dạng khoảng như cổng nguồn và cổng đích vì cần phải xây dựng tập các tiền tố đại diện cho khoảng giá trị đó [15]. Điều này làm tăng chi phí cho tính toán đối với việc xây dựng tập tiền tố và tổng hợp các xung đột cho các luật ở bước cuối.
- FAT có cấu trúc cây phức tạp do mỗi nút phải lưu trữ nhiều thông tin.

- Việc sử dụng *element* cho việc lưu trữ từng byte dẫn đến số lượng các *element* khi chuyển đổi và lưu trữ các luật trong tập luật là rất lớn. Do đó, cấu trúc này yêu cầu chi phí cao về bộ nhớ, cũng như thời gian cho việc xây dựng và thay đổi cây.

III. CẤU TRÚC CÂY PHÁT HIỆN XUNG ĐỘT TRONG TẬP LUẬT TƯỜNG LỬA

Theo lý thuyết thì xung đột giữa hai luật được căn cứ vào 3 yếu tố: mối quan hệ không gian luật giữa chúng, hành động (action) gắn với mỗi luật và thứ tự của luật. Vì hành động và thứ tự của luật là các giá trị tường minh nên việc xác định xung đột giữa hai luật thực chất là việc xác định mối quan hệ không gian luật của chúng. Thuật toán chúng tôi đề xuất gồm cấu trúc CDT và các thủ tục xây dựng cấu trúc CDT từ tập luật của tường lửa nhằm tìm ra mối quan hệ giữa không gian luật của chúng một cách hiệu quả, từ đó xác định các xung đột giữa các luật.

1. Các định nghĩa

1) Mức độ chi tiết của trường:

Trong thực tế, giá trị của các trường của một luật có thể là kiểu tiền tố, một khoảng giá trị hay các một tập các giá trị riêng biệt. Mỗi tiền tố, hay một khoảng giá trị sẽ bao gồm một tập các giá trị thỏa mãn, số lượng các giá trị trong tập này càng ít thì độ chi tiết của trường càng cao. Trong bài báo, chúng tôi sử dụng trọng số này nên xây dựng định nghĩa về nó.

Định nghĩa 1: Độ chi tiết của trường f_n được ký hiệu là $|f_n|$ và được xác định như sau.

Nếu f_n là kiểu tiền tố, độ chi tiết f_n được tính bằng chiều dài của tiền tố đó. Với các trường IP nguồn và IP đích sử dụng địa chỉ IPv4 thì độ chi tiết cao nhất của các trường này là 32.

Nếu f_n là kiểu khoảng giá trị $[a : b]$, độ chi tiết f_n được tính dựa trên số lượng các giá trị trong khoảng đó theo công thức sau:

$$|f_n| = \left(\frac{\text{MAX} - (b-a)}{\text{MAX}}\right) \times L, \tag{1}$$

trong đó MAX là giá trị lớn nhất mà a và b có thể nhận, L là bậc cao nhất mà mức độ chi tiết của f_n có thể đạt.

Để phù hợp trong quá trình so sánh mức độ chi tiết của trường có kiểu là tiền tố với trường có kiểu dữ liệu là khoảng, chúng tôi chọn L là độ dài được tính bằng bít của các số nguyên a và b. Trong các gói tin IPv4 thì các trường cổng nguồn và cổng đích được lưu trong các số nguyên 16 bít nên $L = 16$ và MAX = 65535.

Với trường hợp giá trị của trường f_n là tập gồm n giá trị riêng biệt thì luật có thể được tách ra thành n luật mà giá trị của trường f_n trong mỗi luật là một trong các giá trị của tập ban đầu và khi đó có thể coi đây là một khoảng mà giá trị đầu trùng với giá trị cuối và mức độ chi tiết của trường f_n trong luật này là cao nhất. Ví dụ, luật R có trường địa chỉ nguồn của luật cho dưới dạng tiền tố 000100* thì $|\text{IP}_{\text{source}}| = 6$. Trường cổng đích của R có giá trị là 80 thì có thể coi được cho dưới dạng khoảng giá trị [80 : 80] và $|\text{Port}_{\text{des}}| = 16$.

2) Mối quan hệ giữa hai giá trị trường:

Định nghĩa 2: Mối quan hệ giữa hai giá trị trường V_1 và V_2 của trường f_n.

V_1 **trùng** V_2 (ký hiệu $V_1 \approx V_2$) khi và chỉ khi: Nếu f_n được cho dưới dạng tiền tố thì $V_1 = V_2$; Nếu f_n được cho dưới dạng khoảng giá trị $V_1 = [a : b]$ và $V_2 = [c : d]$ thì $a = c$ và $b = d$.

V_1 **thuộc** V_2 (ký hiệu $V_1 \in V_2$) khi và chỉ khi: Nếu f_n được cho dưới dạng tiền tố thì V_2 là tiền tố của V_1; Nếu f_n được cho dưới dạng khoảng giá trị $V_1 = [a : b]$, $V_2 = [c : d]$ thì ($a \geq c$ và $b < d$) hoặc ($a > c$ và $b \leq d$).

V_1 **giao** V_2 (ký hiệu $V_1 \S V_2$) khi và chỉ khi: Nếu f_n được cho dưới dạng khoảng giá trị $V_1 = [a : b]$, $V_2 = [c : d]$ thì ($a < c \leq b < d$) hoặc ($c < a \leq d < b$).

V_1 **tách rời** V_2 (ký hiệu $V_1 >< V_2$) khi và chỉ khi: Nếu f_n được cho dưới dạng tiền tố thì V_1 không phải là tiền tố của V_2 và ngược lại; Nếu f_n được cho dưới dạng khoảng giá trị $V_1 = [a : b]$, $V_2 = [c : d]$ thì $b < c$ hoặc $a > d$.

Định lý 1: Cho hai giá trị trường V_1 và V_2 của trường f_n, ta có:

(i) Điều kiện cần để $V_1 \in V_2$ là $|V_1| > |V_2|$;
(ii) Điều kiện cần để $V_1 \approx V_2$ là $|V_1| = |V_2|$.

Chứng minh:

(i) Theo định nghĩa của mối quan hệ $V_1 \in V_2$, có hai trường hợp xảy ra. Trường hợp f_n được cho dưới dạng tiền tố, để V_2 là tiền tố của V_1 thì trước hết chuỗi V_1 phải có độ dài lớn hơn chuỗi V_2, viết là $|V_1| = \text{len}(V_1) > \text{len}(V_2) = |V_2|$.

Trường hợp f_n được cho dưới dạng khoảng giá trị $V_1 = [a : b]$ và $V_2 = [c : d]$, khi đó ta có:

$$\begin{aligned} |V_1| - |V_2| &= \left(\frac{\text{MAX} - (b-a)}{\text{MAX}}\right) \times L \\ &\quad - \left(\frac{\text{MAX} - (d-c)}{\text{MAX}}\right) \times L \\ &= L \times \left(\frac{(d-b) + (a-c)}{\text{MAX}}\right). \end{aligned}$$

Theo định nghĩa của $V_1 \in V_2$, ta có ($a \geq c$ và $b < d$) hoặc ($a > c$ và $b \leq d$). Với ($a \geq c$ và $b < d$) thì $(a - c) \geq 0$ và $(d - b) > 0$, từ đó $(d - b) + (a - c) > 0$, suy ra: $|V_1| - |V_2| > 0 \Leftrightarrow |V_1| > |V_2|$. Với ($a > c$ và $d \geq b$) thì

$(a - c) > 0$ và $(d - b) \geq 0$, từ đó $(d - b) + (a - c) > 0$, suy ra: $|V_1| - |V_2| > 0 \Leftrightarrow |V_1| > |V_2|$.

(*ii*) Theo định nghĩa của mối quan hệ trùng giữa hai giá trị trường, dễ dàng thấy được khi $V_1 \approx V_2$ thì $|V_1| = |V_2|$. □

Định lý 2: Cho một tập các giá trị trường $\boldsymbol{V} = (V_1, V_2, \ldots, V_m)$ của trường f_n. Nếu V_k là giá trị trường có độ chi tiết lớn nhất trong tập $\boldsymbol{V}$ thì với mọi $V_i \in \boldsymbol{V}$ $(i \neq k, 1 \leq i \leq m)$ ta luôn có $V_i \notin V_k$.

Chứng minh: Vì V_k có độ chi tiết lớn nhất trong $\boldsymbol{V}$ nên $|V_k| \geq |V_i|$. Mặt khác, theo định lý 1, điều kiện cần để $V_i \in V_k$ là $|V_i| > |V_k|$, nên $V_i \notin V_k$. □

3) Cấu trúc luật:

Cấu trúc lưu trữ giá trị trường của một luật như sau:

```
struct typeofvalue{
   public bool isPrefix;
   public string prefix;
   public int begin;
   public int end;}
```

trong đó, *isPrefix* là 1 nếu dữ liệu trường đang xét được cho dưới dạng tiền tố (source IP, destination IP, protocol), *isPrefix* là 0 nếu dữ liệu trường đang xét không phải dạng tiền tố (source port, destination port), *begin* và *end* là giá trị bắt đầu và kết thúc giá trị của trường (trường hợp *isPrefix* = 0), *prefix* là chuỗi tiền tố của trường (trường hợp *isPrefix* = 1).

Để lưu trữ các luật trong cấu trúc CDT chúng tôi xây dựng lại cấu trúc luật gồm danh sách các dữ liệu trường và *Action*. Mỗi trường được lưu trữ trong một bản ghi *unit* gồm các thông tin:

```
struct unit{
   public int type;
   public float detail;
   public int ruleid;
   public typeofvalue value;}
```

trong đó, *type* là kiểu trường (1: source IP, 2: destination IP, 3: source port, 4: destination port, 5: protocol), *detail* là mức độ chi tiết của trường, được xác định theo định nghĩa 1, *ruleid* là chỉ số luật, *value* là giá trị của trường.

Với cấu trúc như trên, mỗi *unit* sẽ gắn với một trường của một luật cụ thể và có thể lưu trữ được cả dữ liệu dạng tiền tố và dữ liệu dạng khoảng cũng như dạng giá trị đơn lẻ. Ví dụ, luật R có chỉ số là *n* với các tham số $(src\ IP, des\ IP, src\ port, des\ port, Action) = (10.10.0.0/16, 10.0.0.0/8, 80, *, Deny)$ sẽ được biểu diễn bằng danh sách 4 *unit* như trong bảng I.

Luật R sẽ được biểu diễn dưới dạng R = (*unit*[0], *unit*[1], *unit*[2], *unit*[3], *Action*), với giá trị của các *unit* được thể hiện trong bảng I.

Bảng I
DANH SÁCH CÁC UNIT CỦA LUẬT R

	unit[0]	*unit*[1]	*unit*[2]	*unit*[3]
type	1	2	3	4
detail	16	8	16	0
ruleid	n	n	n	n
Value.isPrefix	1	1	0	0
Value.begin			80	0
Value.end			80	65535
Value.prefix	00001010 00001010	0000 1010		

Định nghĩa 3: Cho hai *unit* u_1 và u_2, phép **so sánh** giữa u_1 và u_2 được xác định như sau: $u_1 < u_2$ khi $u_1.detail < u_2.detail$, $u_1 > u_2$ khi $u_1.detail > u_2.detail$, $u_1 = u_2$ khi $u_1.detail = u_2.detail$.

Định nghĩa 4: Các mối quan hệ giữa hai *unit* u_1 và u_2 (chỉ sử dụng khi u_1 và u_2 có cùng kiểu trường), bao gồm: u_1 **trùng** u_2 (ký hiệu $u_1 \approx u_2$) khi và chỉ khi $u_1.value \approx u_2.value$, u_1 **thuộc** u_2 (ký hiệu $u_1 \in u_2$) khi và chỉ khi $u_1.value \in u_2.value$, u_1 **giao** u_2 (ký hiệu $u_1 \S u_2$) khi và chỉ khi $u_1.value \S u_2.value$.

2. Ý tưởng cơ bản của thuật toán

Thuật toán chúng tôi đề xuất bao gồm xây dựng cấu trúc CDT từ các luật nhằm xác định mối quan hệ không gian luật của chúng. Việc xây dựng cấu trúc CDT tuân theo các nguyên tắc chính sau đây:

i) Các luật được chuyển sang cấu trúc gồm các *unit* để làm đầu vào cho quá trình xây dựng cây.

ii) Để xác định mối quan hệ giữa hai không gian luật, chúng ta phải xét mối quan hệ của chúng trong từng chiều trong không gian đó.

iii) Tại mỗi chiều không gian luật, luật nào có mức độ chi tiết cao nhất sẽ được xem xét mối quan hệ với các luật còn lại. Theo lựa chọn này, căn cứ vào định lý 2 thì luật hay nhóm luật đang được xét sẽ chỉ có mối quan hệ với các luật khác thuộc các dạng: Trùng khớp (Match), là tập con (Subset), giao nhau (Overlap), tách biệt (Disjoint). Trong các mối quan hệ đó mối quan hệ tách biệt không tạo ra xung đột nên không cần xem xét. Vì vậy tại một thời điểm, chúng ta chỉ cần quan tâm đến các luật có không gian luật trùng khớp (***Match***), chứa (***Super***), giao nhau (***Overlap***) với không gian luật hay nhóm luật đang xét.

iv) Với một luật đang xét, tại chiều thứ $i + 1$: Tập các luật trùng khớp với nó sẽ được kiểm tra trong tập ***Match*** của chiều thứ i; tập các luật chứa được kiểm tra trong tập ***Match***, ***Super*** của chiều thứ i; tập các

luật giao nhau được kiểm tra trong tập ***Match***, ***Super*** và ***Overlap*** của chiều thứ i. Phép chuyển luật từ các tập như sau:

$$(\textbf{\textit{Match}})_i \xrightarrow{condition\_match} (\textbf{\textit{Match}})_{i+1}, \quad (2)$$

$$(\textbf{\textit{Match}})_i \xrightarrow{condition\_super} (\textbf{\textit{Super}})_{i+1}, \quad (3)$$

$$(\textbf{\textit{Super}})_i \xrightarrow{condition\_super} (\textbf{\textit{Super}})_{i+1}, \quad (4)$$

$$(\textbf{\textit{Match}})_i \xrightarrow{condition\_overlap} (\textbf{\textit{Overlap}})_{i+1}, \quad (5)$$

$$(\textbf{\textit{Super}})_i \xrightarrow{condition\_overlap} (\textbf{\textit{Overlap}})_{i+1}, \quad (6)$$

$$(\textbf{\textit{Overlap}})_i \xrightarrow{condition\_overlap} (\textbf{\textit{Overlap}})_{i+1}, \quad (7)$$

trong đó "*condition_x*" là điều kiện để một luật chuyển từ một tập của bước thứ i sang tập của bước thứ $i+1$. Gọi R là luật đang xét, $R(f_m)$ là giá trị trường thứ m của R, thì điều kiện "*condition_x*" để luật P được chuyển trong các công thức trên như sau:
condition_match: $R(f_{i+1}) \approx P(f_{i+1})$
condition_super: $R(f_{i+1}) \in P(f_{i+1})$
condition_overlap: $R(f_{i+1})$ § $P(f_{i+1})$

v) Mối quan hệ thực sự giữa không gian luật của một luật với các luật khác sẽ được xác định tại bước cuối cùng khi tất cả các chiều đã được kiểm tra.

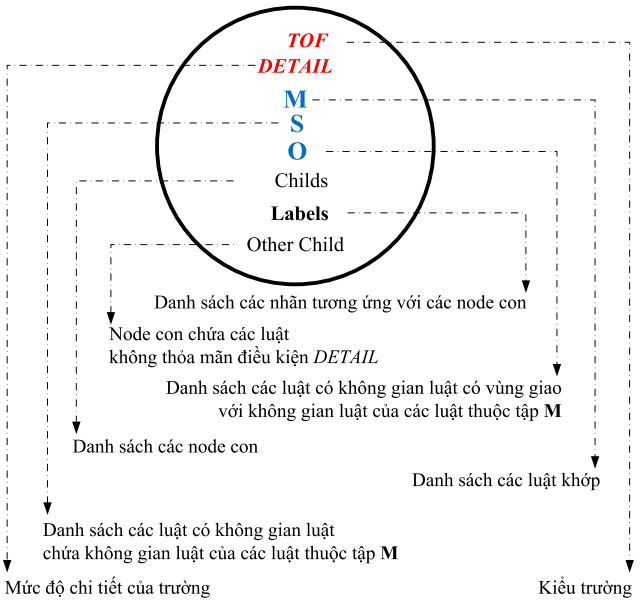
3. Cấu trúc CDT

CDT là cây đa nhánh được xây dựng từ tập dữ liệu đầu vào là các *unit* của tập luật. Nút gốc ROOT của CDT chứa danh sách tất cả các luật trong tập luật. Trong cây, đường dẫn từ nút gốc ROOT đến nút lá biểu diễn hoàn chỉnh một hay một nhóm luật thỏa mãn các điều kiện cụ thể trên đường dẫn đó. Nút N mang thông tin về kiểu trường f_n và mức độ chi tiết của trường đó, các nút con của N được xây dựng theo giá trị trường f_n và độ chi tiết của N luôn lớn hơn hoặc bằng độ chi tiết của trường được lưu trong các nút con của nó.

1) Cấu trúc nút:

Nút của CDT có cấu trúc như hình 2, trong đó:

- *TOF* (Type Of Field) là kiểu trường được thể hiện tại nút. Các quy ước như sau: 1 là source IP, 2 là destination IP, 3 là source port, 4 là destination port, 5 là protocol.
- *DETAIL* là mức độ chi tiết của trường;
- **M** là danh sách các luật thỏa mãn điều kiện đường dẫn từ ROOT đến nút hiện tại;
- **S** là danh sách các luật có không gian luật chứa không gian luật của các luật thuộc **M**;
- **O** là danh sách các luật có không gian luật có một phần chung với không gian luật của các luật thuộc **M**;
- **Labels** là tập các nhãn được gán cho các nút con của N;



Hình 2. Cấu trúc nút của CDT.

- *Childs* là danh sách các nút con của nút. Nút con thứ i sẽ chứa các luật thuộc **M** thỏa mãn điều kiện trường thứ [*TOF*] có độ chi tiết [*DETAIL*] và có giá trị là **Labels**[i];
- OtherChild là nút con đặc biệt của nút đang xét, chứa các luật thuộc **M** không thỏa mãn điều kiện: Trường *TOF* có mức độ chi tiết bằng *DETAIL*.

2) Thủ tục xây dựng nút:

Thủ tục xây dựng nút được trình bày trong thuật toán 1. Nút N được xây dựng với đầu vào gồm ba tập *unit*: **Unit-matchs** chứa các *unit* của các luật khớp với đường dẫn từ ROOT tới N; **Unit-Supers** chứa các *unit* của các luật có không gian luật chứa không gian luật thỏa mãn điều kiện đường dẫn từ ROOT tới N; **Unit-Overlaps** chứa các *unit* của các luật có không gian luật có một phần chung với không gian luật thỏa mãn điều kiện đường dẫn từ ROOT tới N.

Các giá trị *TOF* và *DETAIL* được đặt như sau:
$N.TOF = UMAX.type$,
$N.DETAIL = UMAX.detail$,
trong đó $UMAX \in$ **Unit-matchs** thỏa mãn với mọi $u \in$ **Unit-matchs** thì $u.detail \leq UMAX.detail$.

Các tập **M**, **S** và **O** được lấy chỉ số các luật trong ba tập tương ứng **Unit-matchs**, **Unit-Supers** và **Unit-Overlaps**. Ví dụ, trong tập **Unit-matchs** có các *unit* của các luật số 1, 7, 9 thì **M** sẽ bao gồm các số 1, 7, 9.

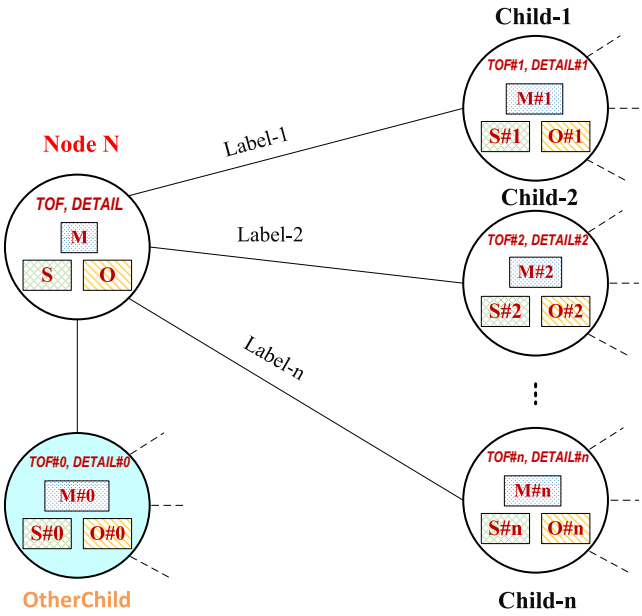
Labels: Gọi **lstUnit** là tập các *unit* thuộc **Unit-matchs** có kiểu trường bằng $N.TOF$ và độ chi tiết bằng $N.DETAIL$. Nếu mô phỏng bằng câu lệnh SQL thì **lstUnit** được trả về từ truy vấn "SELECT FROM **Unit-matchs** WHERE (*type* =

Thuật toán 1: BuildNode

Input: List of unit **UnitMats**;
List of unit **UnitSups**;
List of unit **UnitOvers**;
Output: CDTNode *N*
Begin
1: *UMAX* = GetMaxUnit(**UnitMats**);
2: **lstUnit** = GetUnits(*UMAX*, **UnitMats**);
3: *N.TOF* = *UMAX.type*;
4: *N.DETAIL* = *UMAX.detail*;
5: **for each** *u* of **lstUnit**
6: *ulabel*= CreateLabel(*u*);
7: **if** *ulabel* **not in** N.Labels
8: N.Labels.add(*ulabel*);
9: (**UnitMats**) $\xrightarrow{condition\_match}$ (**uMatchs**);
10: (**UnitMats**) $\xrightarrow{condition\_super}$ (**uSupers**);
11: (**UnitSups**) $\xrightarrow{condition\_super}$ (**uSupers**);
12: (**UnitMats**) $\xrightarrow{condition\_overlap}$ (**uOverlaps**);
13: (**UnitSups**) $\xrightarrow{condition\_overlap}$ (**uOverlaps**);
14: (**UnitOvers**) $\xrightarrow{condition\_overlap}$ (**uOverlaps**);
15: CDTNode *M*;
16: BuildNode(M, **uMatchs**, **uSupers**, **uOverlaps**);
17: N.Childs.add(*M*);
18: RemoveUnit(**UnitMats**, **uMatchs**);
19: **end if**
20: **end for each**
21: BuildNode(*N.OtherChild*, **UnitMats**, **UnitSups**, **UnitOvers**);
End

Thuật toán 2: BuildTree

Input: *Rule Set*;
Output: CDTTree CDT;
Begin
1: **uMatchs** =GetUnitFromRuleSet(*Rule Set*);
2: **uSupers** = ∅;
3: **uOverlaps** = ∅;
4: BuildNode(ROOT, **uMatchs**, **uSupers**, **uOverlaps**);
End



Hình 3. Nút *N* sau khi được xây dựng.

N.TOF AND *detail* = *N.DETAIL*)". Trong trường hợp các *unit* thuộc **lstUnit** là kiểu tiền tố thì **Labels** là tập các tiền tố không trùng nhau của các *unit* thuộc **lstUnit**, trường hợp dữ liệu là kiểu khoảng thì **Labels** là tập các khoảng giá trị không trùng nhau của các *unit* thuộc **lstUnit**.

Childs: Nút con thứ *i* được xây dựng với bộ ba đầu vào được xác định theo các phép chuyển luật (2), (3), (4), (5), (6), (7) với giá trị phân loại là **Labels**(*i*).

OtherChild: Là nút đặc biệt của *N*. Tất cả các luật thuộc tập **Unit-matchs** có kiểu trường *N.TOF* mà độ chi tiết nhỏ hơn *N.DETAIL* là đầu vào để xây dựng nút đặc biệt. Tập **S** và **O** của OtherChild chính là tập **S** và **O** của *N*.

Dòng 1: Hàm GetMaxUnit trả về *unit* - *UMAX* có độ chi tiết lớn nhất trong tập các **UnitMats**.

Dòng 2: Hàm GetUnits trả về các *unit* trong **UnitMats** có cùng *type* và cùng *detail* với *UMAX*.

Dòng 3, 4: Gán các giá trị *TOF* và *DETAIL* cho nút *N*.

Dòng 6: Hàm CreateLabel(*u*) tạo nhãn tạm từ *u.value*, nhãn này được sử dụng để kiểm tra *u.value* đã được sử dụng tạo ra cây con của *N* hay chưa tại dòng 8.

Dòng 8: Thêm nhãn tạm vào danh sách.

Dòng 9, 10, 11, 12, 13, 14: Xây dựng các tập đầu vào cho cây con mới của *N*.

Dòng 15, 16, 17: Thêm cây con mới cho *N*.

Dòng 18: Loại các luật đã xét khỏi tập **UnitMats**.

Dòng 21: Xây dựng nút đặc biệt *N.OtherChild*.

Nút *N* sau khi được xây dựng có cấu trúc như hình 3.

3) Thủ tục xây dựng cây:

Thủ tục xây dựng cây được trình bày trong thuật toán 2.

Dòng 1: Đọc và chuyển tất cả các luật sang các *unit*, lưu trữ tất cả vào **uMatchs**.

Dòng 2, 3: Khởi tạo các tập **uSupers**, **uOverlaps**.

Dòng 4: Xây dựng CDT bắt đầu từ nút gốc ROOT.

Minh họa cho thuật toán 1 và thuật toán 2, chúng tôi thực hiện xây dựng CDT cho tập luật ở bảng II, trong đó, mỗi luật trong tập luật chỉ bao gồm 4 chiều và các quy ước như sau: **R** là Rule, **S-IP** là Source IP address,

Bảng II
TẬP LUẬT 4 CHIỀU

R	S-IP	D-IP	S-P	D-P	Act
R1	192.168.0.0/23	*	80:110	*	Acc
R2	192.168.10.0/23	10.0.0.0/9	80	*	Acc
R3	193.0.0.0/8	10.0.0.0/8	*	*	Acc
R4	*	10.10.0.0/16	80	*	Deny
R5	192.168.1.1/32	10.0.0.0/8	80	*	Acc

Bảng III
ĐỘ CHI TIẾT CỦA CÁC GIÁ TRỊ TRƯỜNG

Value	Prefix	Detail
192.168.1.1/32	P1=1100000010101000 0000000100000001	32
192.168.0.0/23	P2=110000001010100000000000	23
192.168.10.0/23	P3=110000001010100000000101	23
10.10.0.0/16	P4=0000101000001010	16
80		16
80:110		15,99
10.0.0.0/9	P5=000010100	9
10.0.0.0/8	P6=00001010	8
193.0.0.0/8	P7=11000001	8
*		0

D-IP là Destination IP address, **S-P** là Source port, **D-P** là Destination port, **Act** là Action.

Việc tính toán mức độ chi tiết của các giá trị trường và tính toán chuỗi tiền tố trong các *unit* của tập luật có trong bảng II thể hiện tại bảng III.

Để dễ quan sát, chúng tôi ký hiệu một *unit* được thể hiện theo định dạng sau: $[ruleid; type; detail]$ các giá trị khác không được thể hiện.

R1=[1;1;23], [1;3;15,99], [1;2;0],[1;4;0]

R2=[2;1;23], [2;3;16], [2;2;9],[2;4;0]

R3=[3;1;8], [3;2;8], [3;3;0],[3;4;0]

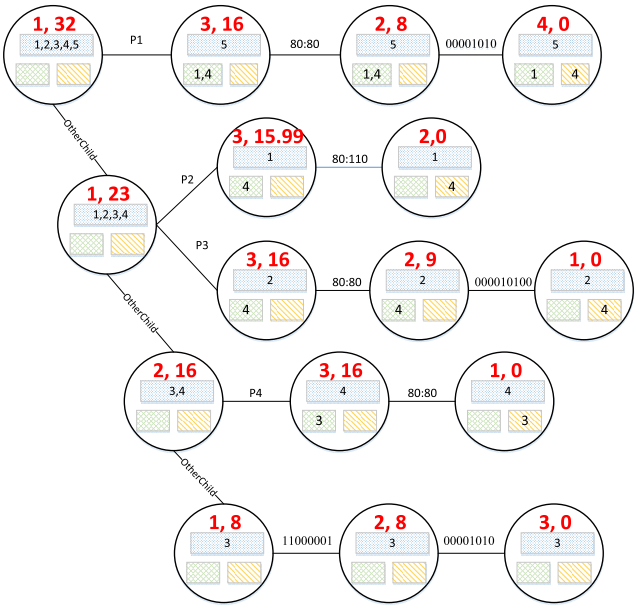
R4=[4;2;16], [4;3;16], [4;1;0],[4;4;0]

R5=[5;1;32], [5;3;16], [5;2;8],[5;4;0]

Các nút được biểu diễn theo định dạng $[TOF; DETAIL]$.

CDT được xây dựng từ tập các *unit* của các luật trên. Quá trình xây dựng cây như sau:

Nút gốc ROOT được xây dựng với đầu vào là tất cả các *unit*: $ROOT.M = (1..5)$, $ROOT.S = ROOT.O = \emptyset$. *Unit* có độ chi tiết lớn nhất trong danh sách $ROOT.M$ là $[5;1;32]$, nên ROOT = $[1;32]$. Nút $[1,32]$ chỉ có 1 nút con $N1$, $N1.M = (5)$, nhãn của nhánh là P1. Vì R1, R4 có trường địa chỉ nguồn là tiền tố của R5, nên $N1.S = (1,4)$, $N1.O = \emptyset$. Các *unit* còn lại của $N1.M$ là $[5;3;16]$, $[5;2;8]$,



Hình 4. CDT của tập luật bảng II.

$[5;4;0]$ nên $N1 = [3;16]$. Quá trình tiếp theo tương tự $N1$ có một con là $N2 = [2;8]$. $N2$ có nút con là $N3 = [4,0]$. Tại $N2$ do R5.DesIPAddr là tiền tố của R4.DesIPAddr nên $N3$ có $N3.O = (4)$.

Nhánh $ROOT.OtherChild$ được xây dựng với tập đầu vào là các luật (1..4). Quá trình xây dựng tương tự như ở nút ROOT.

Chú ý: Để tối ưu hóa quá trình lưu trữ, trên đường dẫn luật, nếu gặp nút có độ chi tiết bằng 0 quá trình phát triển các nút con sẽ kết thúc và các luật trong tập **S** nếu còn các *unit* có độ chi tiết khác không sẽ được chuyển sang tập **O**. Trường hợp này trong cây là đường dẫn luật của R1, R4, R3.

4) Thủ tục chèn luật mới:

Thủ tục chèn luật mới được sử dụng nhằm thay đổi nội dung CDT mà không cần phải xây dựng lại toàn bộ cây trong trường hợp thay đổi chính sách an ninh trên tường lửa từ đó cần thêm mới các luật vào tập luật.

Chèn luật mới vào CDT được thực hiện bằng việc chèn danh sách các *unit* của luật đó vào nút gốc. Thuật toán chèn (thuật toán 3) được chúng tôi xây dựng nhằm thực hiện chèn một danh sách các *unit* vào nút N.

Gọi *UnitsOfRule* là danh sách các *unit* của luật R cần chèn, *UMAX* = GetMaxUnit(*unitsOfRule*) và k là chỉ số của luật cần chèn, các trường hợp có thể xảy ra như sau.

Trường hợp 1: N là nút rỗng.

Trong trường hợp này việc chèn các *unit* vào N chính là việc xây dựng nút mới với đầu vào là *unitsOfRule*. Thao tác chèn được minh họa trong hình 5(a).

Trường hợp 2: Độ chi tiết của *UMAX* lớn hơn *N.DETAIL*.

Trong trường hợp này, độ chi tiết của *unit* lớn nhất trong R lớn hơn độ chi tiết của *N*:

- Xây dựng nút mới với đầu vào là ba tập như sau:
  matchs = GetUnitFromID($N.M$) ∪ *unitsOfRule*,
  supers = GetUnitFromID($N.S$),
  overlaps = GetUnitFromID($N.O$),
  trong đó, hàm GetUnitFromID thực hiện lấy các *unit* từ tập các chỉ số luật. Trong quá trình xây dựng nút mới bỏ qua bước xây dựng OtherChild.
- *N* chuyển thành OtherChild của nút mới

Thao tác chèn được minh họa trong hình 5(b).

Trường hợp 3: Độ chi tiết của *UMAX* bằng *N.DETAIL*.

a) Trường hợp *UMAX.type* = *N.TOF*
Tính nhãn tạm *ulabel* = CreateLabel(*UMAX*)
Trường hợp *ulabel* = *N*.**Labels**[*index*]
Giá trị của *UMAX* thuộc cây con thứ *index* của nút *N*, các thao tác được thực hiện như sau:
  + Thêm chỉ số luật của R vào N.M.
  + Trong trường hợp *UMAX* được cho dưới dạng khoảng thì *UMAX* vẫn có khả năng có quan hệ giao nhau với các khoảng giá trị khác cùng mức độ chi tiết nên phải cập nhật các thông tin các trường hợp Giao nhau giữa *UMAX* với các nút con từ $N.Childs[0]$ đến $N.Childs[index-1]$.
  + Xây dựng lại nút con $N.Childs[index]$.

  Thao tác chèn được minh họa trong hình 5(c).
  Trường hợp *ulabel* ≠ *N*.**Labels**[*index*], ∀ *index*
  Giá trị của *UMAX* không thuộc tập các cây con của nút *N*, các thao tác được thực hiện như sau:
  + Thêm chỉ số luật của R vào N.M.
  + Cập nhật các thông tin các trường hợp Giao nhau giữa *UMAX* với các nút con của *N*.
  + Xây dựng thêm nút con mới của *N*.

  Thao tác chèn được minh họa trong hình 5(d).

b) Trường hợp *UMAX.type* ≠ *N.TOF*
Trong trường hợp này trường *N.TOF* của luật R có độ chi tiết nhỏ hơn *N.DETAIL*. Các thao tác được thực hiện như sau:
  + Thêm chỉ số luật của R vào N.M.
  + Cập nhật các tập **Supers** và **Overlaps** của các nút của *N*.
  + Chèn luật tập *unitsOfRule* vào *N.OtherChild*.

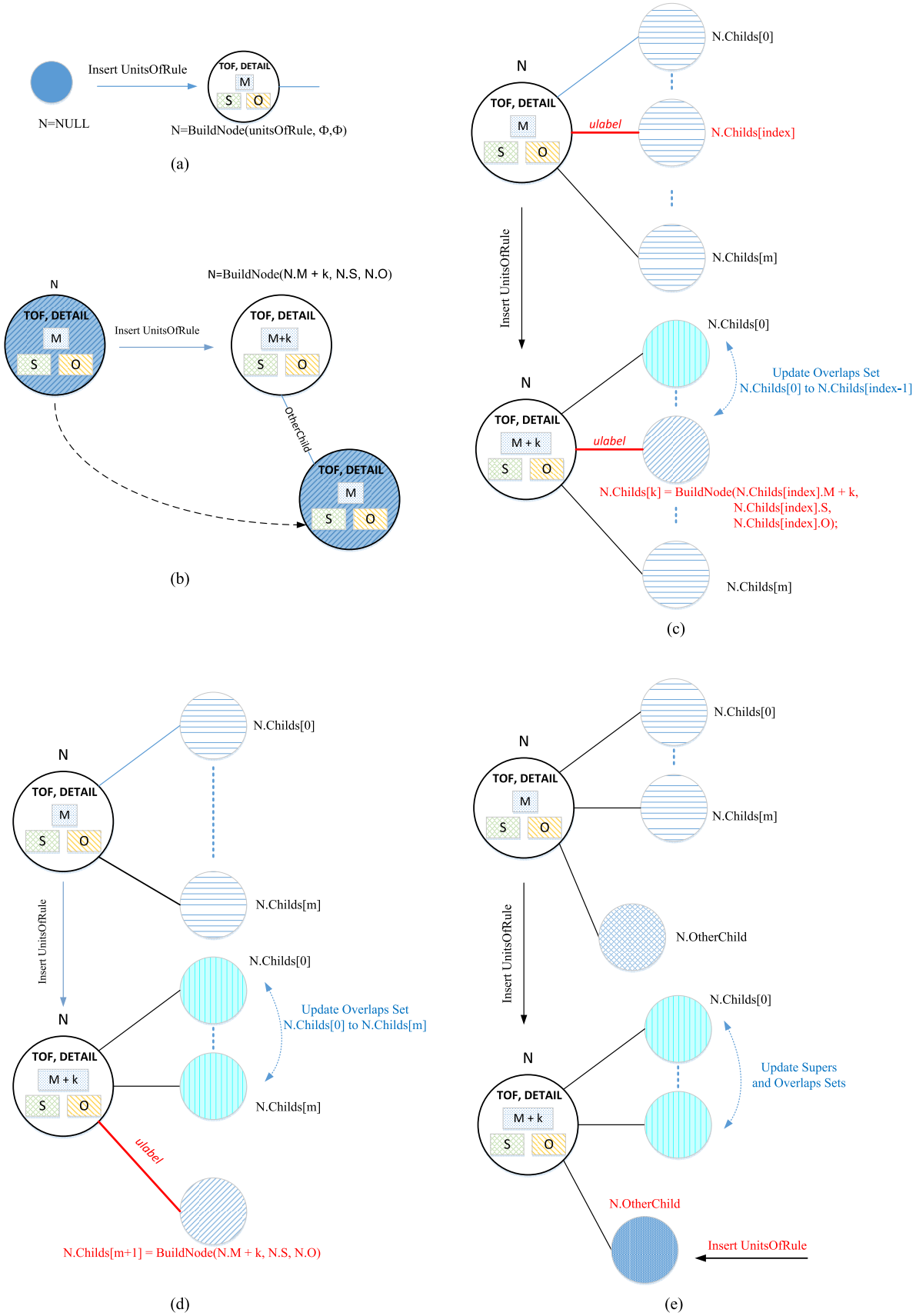
  Thao tác chèn được minh họa trong hình 5(e).

Trường hợp 4: Độ chi tiết của $UMAX$ nhỏ hơn *N.DETAIL*.

Trường hợp này bản chất giống như trường hợp 3-b), các thao tác thực hiện được mô tả trong hình 5(e).

Thuật toán 3: InsertRuleToNode

```
Input: Units of Rule: unitsOfRule; CDTNode N;
Output: CDTNode N;
Begin
1:  UMAX = GetMaxUnit(unitsOfRule);
2:  ulabel = CreateLabel(u);
    //Trường hợp 5a
3:  if (N = NULL) then
4:      N = BuildNode(unitsOfRule, ϕ, ϕ);
5:      return;
6:  end if
    //Trường hợp 5b
7:  if (UMAX.detail > N.DETAIL) then
8:      X = BuildNode(N.M + k, N.S, N.O);
9:      X.OtherChild = N;
10:     N = X;
11:     return;
12: end if
    //Trường hợp 5c, 5d
13: if (UMAX.detail = N.DETAIL)
    and (UMAX.type = N.TOF) then
14:     index = N.Labels.Find(ulabel);
    //Trường hợp 5c
15:     if (index >= 0) then
16:         for (i = 0; i < index; index++) do
17:             UpdateOverlaps(N.Childs[i]);
18:         end for
19:         N.Childs[index] = BuildNode
                (N.Childs[index].M + k,
                N.Childs[index].S, N.Childs[index].O);
20:         return;
21:     end if
    //Trường hợp 5d
22:     if (index < 0) then
23:         for each u in N.Childs do
                UpdateOverlaps(u);
24:         end for
25:         X = BuildNode(N.M + k, N.S, N.O);
26:         N.Childs.add(X);
27:         return;
28:     end if
29: end if
    //Trường hợp 5e
30: for each u in N.Childs do
        UpdateSuperAndOverlaps(u);
31: end for each
32: InsertRuleToNode(unitsOfRule, N.OtherChild);
End
```



Hình 5. Chèn luật mới vào nút N.

Thuật toán 4: DeleteRuleFromNode

Input: Index Of Rule: k; CDTNode N;
Output: CDTNode N;
Begin
1: **if** ($N = NULL$) **then**
2: **return**;
3: **end if**
4: N.M.Remove(k);
5: N.S.Remove(k);
6: N.O.Remove(k);
7: **if** ($N.M.count = 0$) **then**
8: $N = NULL$;
9: **return**;
10: **end if**
11: **for each M in** $N.Childs$ **do**
DeleteRuleFromNode(k, **M**);
12: **end for each**
13: DeleteRuleFromNode($k, N.OtherChild$);
End

5) Thủ tục xóa luật:

Cùng giống như việc thủ tục thêm luật mới, thủ tục xóa luật (thuật toán 4) được sử dụng nhằm thay đổi nội dung CDT mà không cần phải xây dựng lại toàn bộ cây trong trường hợp thay đổi chính sách an ninh trên tường lửa từ đó cần xóa một hay nhiều luật trong tập luật.

Luật R có chỉ số k được xóa khỏi CDT bằng cách loại bỏ tất cả chỉ số k khỏi các nút thuộc CDT. Tại nút N, việc loại bỏ chỉ số k được thực hiện trong cả ba tập **M**, **S** và **O** của N và các nút con của N.

- Dòng 4, 5, 6: Xóa k khỏi các tập **M**, **S**, **O** của N.
- Trong trường hợp N.M chứa duy nhất chỉ số k có nghĩa N là nút chỉ khớp duy nhất với luật R thì khi xóa R cũng tương ứng N bị xóa. Các dòng từ 5 đến 9 thực hiện tác vụ này.
- Dòng 11: xóa k khỏi các cây con của N.
- Dòng 13: xóa k khỏi nút con OtherChild của N.

4. Phát hiện xung đột trên CDT

Thông tin mối quan hệ về không gian luật giữa các luật được chứa trong tất cả các nút lá của cây. Cụ thể, tại nút lá N: Các luật thuộc tập N.M là có không gian luật trùng nhau; Các luật thuộc tập N.S có không gian luật chứa không gian luật của các luật thuộc N.M; Các luật thuộc tập N.O có không gian luật giao với không gian luật của các luật thuộc tập N.M.

Loại xung đột được xác định theo thuật toán 5, cụ thể như sau:

- Trong tập **M**, một luật R trong **M** được kiểm tra với các luật P còn lại trong **M**. Nếu R và P có cùng hành động thì: R là dư thừa (khi R đứng sau P), P là dư thừa (khi R đứng trước P); Nếu R và P có hành động khác thì: R là bóng của P (khi R đứng sau P), P là bóng của R (khi R đứng trước P);
- Trong tập **S**, tất cả các luật P trong **S** được kiểm tra lần lượt với mỗi luật R trong **M**. Nếu R và P có cùng hành động thì: R là dư thừa (khi R đứng sau P), P là dư thừa (khi R đứng trước P). Nếu R và P có hành động khác thì: R là bóng của P (khi R đứng sau P), P là bóng của R (khi R đứng trước P);
- Trong tập **O**, tất cả các luật P trong **O** được kiểm tra lần lượt mỗi luật R trong **M**. Nếu R và P khác hành động thì chúng xung đột với nhau theo kiểu tương quan.

Trong thuật toán 5, ký hiệu $R.order$ là chỉ thứ tự của luật R trong tập luật, ListOfAnomaly là đầu ra của thuật toán và là danh sách các xung đột được phát hiện trong tập luật và thủ tục ListOfAnomaly.add() thực hiện thêm xung đột vừa phát hiện vào danh sách.

Phát hiện xung đột là một vấn đề phức tạp, nhưng giải quyết dứt điểm các xung đột đó là một vấn đề khó khăn hơn rất nhiều. Điều này được chỉ ra trong các công trình [10–14]. Trong phạm vi của bài báo, chúng tôi không đi sâu về phát triển phương pháp xử lý xung đột một cách tổng quan mà chỉ đưa ra các phương án lựa chọn cho người quản trị hệ thống theo nguyên tắc xem xét bảo đảm an ninh hệ thống một cách tốt nhất. Các nguyên tắc xử lý là cách ứng xử với từng loại xung đột:

- Xung đột dư thừa: Luật dư thừa bị xóa bỏ.
- Xung đột bóng: Loại xung đột này thuộc hai dạng cho phép các gói tin đã bị cấm bởi luật trước đó hay ngược lại cấm các gói tin đã được cho phép trước đó. Đây là sự nhầm lẫn trong cấu hình và cần phải kiểm tra chính sách an ninh, trong trường hợp cho phép thì xóa bỏ luật bóng hay thay đổi lại luật đường trước. Trong [14], các tác giả đề xuất thay đổi vị trí của hai luật tuy nhiên điều này vẫn không giải quyết hết xung đột.
- Xung đột tương quan: Có thể sử dụng giải pháp được đề xuất trong [12, 13], tạo một luật mới có không gian luật là phần giao nhau và hành động của luật này phải được quyết định bởi người quản trị và luật này được đặt trước các luật bị giao nhau; Các luật có không gian giao nhau được có thể điều chỉnh cắt bỏ phần không gian chung đó. Hoặc có thể thay đổi thứ tự luật để các gói tin thỏa mãn phần không gian chung được quyết định chính xác theo chính sách an ninh.

IV. CÀI ĐẶT VÀ ĐÁNH GIÁ

Với mục đích kiểm chứng tính đúng đắn và đánh giá cấu trúc CDT, chúng tôi triển khai cài đặt và kiểm thử cấu trúc CDT và FAT [14] là kỹ thuật được đề xuất mới nhất và

Thuật toán 5: DetectAnomaly

```
    Input: CDTNode N;
    Output: List Of Anomaly;
    Begin
 1: if (N = NULL) then
 2:     return;
 3: end if
 4: if (N is leaf) then
    //Duyệt các luật trong N.M
 5:     for each R in N.M do
 6:         for each (P ≠ R) in N.M do
 7:             if (R.Action = P.Action) then
 8:                 if (R.order < P.order) then
 9:                 P is Redanduncy of R;
10:                 else
11:                 R is Redanduncy of P
12:                 ListOfAnomaly.Add();
13:                 end if
14:             else
15:                 if (R.order < P.order) then
16:                 P is Shadowing of R;
17:                 else
18:                 R is Shadowing of P;
19:                 ListOfAnomaly.Add();
20:                 end if
21:             end if
22:         end for each
23:     end for each
    //Duyệt các luật trong N.S
24:     for each P in N.S do
25:         for each R in N.M do
26:             if (R.Action = P.Action) then
27:                 if (R.order < P.order) then
28:                 P is Redanduncy of R;
29:                 else
30:                 R is Redanduncy of P;
31:                 ListOfAnomaly.Add();
32:                 end if
33:             else
34:                 if (R.order < P.order) then
35:                 P is Shadowing of R;
36:                 else
37:                 R is Shadowing of P;
38:                 ListOfAnomaly.Add();
39:                 end if
40:             end if
41:         end for each
42:     end for each
    //Duyệt các luật trong N.O
43:     for each P in N.O do
44:         for each R in N.M do
45:             if (R.Action ≠ P.Action) then
46:             R and P is Correlation
47:             ListOfAnomaly.Add();
48:             end if
49:         end for each
50:     end for each
51: else // Not (N is leaf)
52:     for (i = 0; i < N.Childs.count; i + +) do
53:     DetectAnomaly(N.Childs[i]);
54:     DetectAnomaly(N.OtherChild);
55:     end for
56: end if
    End
```

có nhiều điểm tương đồng. Ngôn ngữ sử dụng cài đặt các thuật toán là Visual *C#* 2013. Quá trình kiểm thử trên nền tảng hệ điều hành Windows 10, 64 bit, phần cứng CPU Intel Core i3 - 4010U, 1,7 GHz, 2 cores, 4 GB RAM.

Để đảm bảo sát với ứng dụng thực tế, chương trình sử dụng các bộ dữ liệu nhân tạo được tạo bằng bộ công cụ ClassBench do Taylor, Turner thuộc Phòng Nghiên cứu ứng dụng, Khoa khoa học máy tính, Đại học Washington, Saint Louis tạo ra [http://www.arl.wustl.edu/classbench]. Các bộ dữ liệu bao gồm các tập luật và các tập tham số gói tin được sinh bởi bộ công cụ trên có dữ liệu đầu vào là những bộ dữ liệu thực của các nhà cung cấp dịch vụ Internet. Đây là bộ công được cộng đồng nghiên cứu sử dụng để đánh giá các thuật toán và các thiết bị phân loại gói tin. Trong quá trình so sánh, vì FAT không thực hiện được với các trường cổng nguồn và cổng đích cho dưới dạng một khoảng giá trị nên chúng tôi chỉ lấy giá trị đầu tiên để có thể kiểm thử với FAT. Ví dụ, trường cổng nguồn là 80:110 thì giá trị trường này khi kiểm tra là 80 với FAT và là khoảng [80:80] với CDT.

1. Kiểm tra khả năng phát hiện xung đột

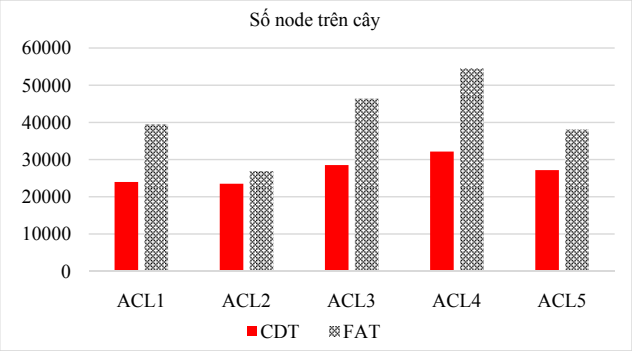
Quá trình kiểm tra khả năng phát hiện xung đột của hai thuật toán sử dụng các cấu trúc FAT và CDT được thử nghiệm trên các tập dữ liệu của Classbench. Kết quả cho thấy số lượng xung đột được phát hiện trên hai cấu trúc là như nhau và được thể hiện trong bảng IV.

2. So sánh về số nút trong cây

Về mặt trực quan, chúng ta dễ dàng nhận thấy trong cấu trúc FAT, mỗi nút sẽ lưu thông tin liên quan đến một byte giá trị thuộc một trường cụ thể. Với cấu trúc này và trong trường địa chỉ là IPv4, số nút tối đa là 4 cho một địa chỉ. Trong cấu trúc CDT mỗi nút sẽ lưu thông tin của một giá

Bảng IV
KẾT QUẢ PHÁT HIỆN XUNG ĐỘT
TRÊN CÁC TẬP LUẬT KHÁC NHAU

Tập luật	Số luật	Số xung đột	
		FAT	CDT
ACL1	17584	222396	222396
ACL2	16194	1948040	1948040
ACL3	18530	542839	542839
ACL4	19086	360136	360136
ACL5	15194	7501	7501



Hình 6. So sánh số lượng nút trên CDT và cây FAT.

trị trường. Việc thử nghiệm với tập luật ví dụ tại bảng II trong [14] thì cấu trúc cây FAT gồm 74 nút, cấu trúc CDT chỉ bao gồm 30 nút và số lượng các xung đột được phát hiện ở hai cây đều là 22. Kết quả thử nghiệm với dữ liệu thực tế, số nút trên FAT và CDT được thể hiện trong hình 6.

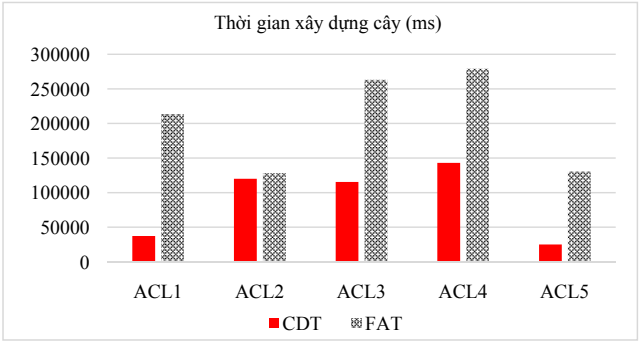
3. So sánh về thời gian xây dựng cây

Quá trình xây dựng các cấu trúc FAT và CDT gồm các bước chính sau đây:

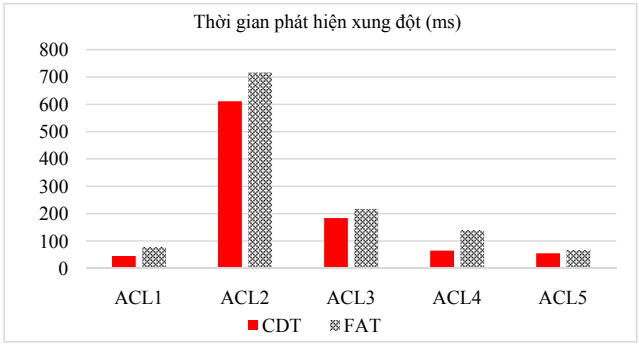
- Chuyển đổi từ tập luật nguyên thủy sang tập các *element* (FAT) và các *unit* (CDT);
- Chèn dữ liệu luật vào cây các *element* (FAT) và các *unit* (CDT).

Quá trình chèn dữ liệu luật sẽ bao gồm thời gian xây dựng của tất cả các nút. Trong quá trình xây dựng một nút, thời gian tính toán cho quá trình phân nhánh tới các nút con là nhiều nhất mà trong đó việc tính toán cho các ứng viên của các tập đầu vào (Primary, Secondary, Tertiary với FAT; ***Match***, ***Super***, ***Overlap*** với CDT) chiếm thời gian chủ yếu. Trong cùng một tập luật, số lượng các *unit* của CDT ít hơn về số lượng các *element* của FAT nên chi phí cho việc lựa chọn các ứng viên của các tập đầu vào trên CDT là nhỏ hơn trên FAT.

Kết quả thực nghiệm tại hình 7, cho thấy thời gian xây dựng CDT nhỏ hơn hơn thời gian xây dựng FAT.



Hình 7. So sánh thời gian xây dựng CDT và FAT.



Hình 8. So sánh thời gian phát hiện xung đột.

4. So sánh thời gian phát hiện xung đột

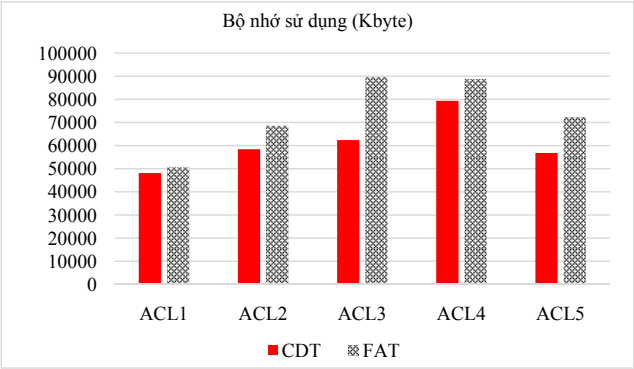
Vì số lượng nút trên cấu trúc CDT ít hơn số nút trên cấu trúc FAT nên thời gian di chuyển từ nút gốc đến nút lá nhằm phát hiện xung đột giữa các luật trên CDT nhỏ hơn so với trên FAT. Kết quả thực nghiệm cũng cho kết quả là trên CDT thời gian phát hiện xung đột nhỏ hơn trên FAT (hình 8).

5. So sánh bộ nhớ lưu trữ

Bộ nhớ sử dụng bởi các cấu trúc CDT và FAT phụ thuộc vào cấu trúc của mỗi nút và tổng số lượng các nút trên cây khi biểu diễn một tập luật. Kết quả thực nghiệm trong hình 9, cho thấy CDT sử dụng bộ nhớ hiệu quả hơn FAT.

V. KẾT LUẬN

Bài báo đã có nghiên cứu và đánh giá về các kỹ thuật phát hiện và xử lý các xung đột trên tập luật nhiều chiều của tưởng lửa. Trên cơ sở chỉ ra hạn chế của các kỹ thuật đã có, bài báo đã đề xuất cấu trúc CDT nhằm phát hiện các xung đột dựa trên việc xác định có hiệu quả mối quan hệ không gian luật giữa các luật trên tường lửa. Cấu trúc CDT được xây dựng trên cơ sở chứng minh về lý thuyết. Các loại xung đột được phát hiện trên CDT chính xác về kiểu và đầy đủ về số lượng.



Hình 9. So sánh bộ nhớ sử dụng.

Kết quả thử nghiệm cho thấy khi so sánh với cấu trúc FAT thì hiệu quả của cấu trúc CDT được thể hiện trên các mặt:

- Số lượng nút trên cây giảm: 13% – 41%;
- Thời gian xây dựng cây giảm: 6% – 83%;
- Thời gian phát hiện xung đột giảm: 15% – 54%;
- Bộ nhớ sử dụng giảm: 5% – 30%.

Cấu trúc CDT hoàn toàn có thể được triển khai trong thực tế với mục đích xây dựng bộ công cụ nhằm phát hiện và xử lý xung đột trong tập luật của các thiết bị tường lửa. Đây là hướng nghiên cứu tiếp theo của chúng tôi.

TÀI LIỆU THAM KHẢO

[1] E. S. Al-Shaer and H. H. Hamed, "Modeling and management of firewall policies," *IEEE Transactions on Network and Service Management*, vol. 1, no. 1, pp. 2–10, 2004.

[2] A. Hari, S. Suri, and G. Parulkar, "Detecting and resolving packet filter conflicts," in *Proceedings of the Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2000)*, vol. 3, 2000, pp. 1203–1212.

[3] H. Lu and S. Sahni, "Conflict detection and resolution in two-dimensional prefix router tables," *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp. 1353–1363, 2005.

[4] A. Kwok and C. K. Poon, "Two-dimensional packet classification and filter conflict resolution in the internet," *Theory of Computing Systems*, vol. 44, no. 3, pp. 289–303, 2009.

[5] C. Maindorfer, "Algorithms and data structures for IP lookup, packet classification and conflict detection," Ph.D. dissertation, University of Freiburg, Germany, 2009.

[6] S. Thanasegaran, Y. Yin, Y. Tateiwa, Y. Katayama, and N. Takahashi, "A topology-based conflict detection system for firewall policies using bit-vector-based spatial calculus," *International Journal of Communications, Network and System Sciences*, vol. 4, no. 11, pp. 683–695, 2011.

[7] C.-L. Lee, G.-Y. Lin, and Y.-C. Chen, "An efficient conflict detection algorithm for packet filters," *IEICE Transactions on Information and Systems*, vol. 95, no. 2, pp. 472–479, 2012.

[8] C.-Y. Lai and P.-C. Wang, "Fast and complete conflict detection for packet classifiers," *IEEE Systems Journal*, vol. 11, no. 2, pp. 1137–1148, 2017.

[9] Vũ Duy Nhất and Nguyễn Mạnh Hùng, "Đề xuất thuật toán phát hiện xung đột giữa các luật hai chiều trong các thiết bị mạng," in *Kỷ yếu Hội thảo Quốc gia lần thứ XIX: Một số vấn đề chọn lọc của Công nghệ Thông tin và Truyền thông*, Oct. 2016.

[10] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 10, pp. 2069–2084, 2005.

[11] L. Yuan, H. Chen, J. Mai, C.-N. Chuah, Z. Su, and P. Mohapatra, "Fireman: A toolkit for firewall modeling and analysis," in *IEEE Symposium on Security and Privacy (S&P'06)*. IEEE, 2006, pp. 15–pp.

[12] M. Abedin, S. Nessa, L. Khan, and B. Thuraisingham, "Detection and resolution of anomalies in firewall policy rules," in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2006, pp. 15–29.

[13] H. Hu, G.-J. Ahn, and K. Kulkarni, "Detecting and resolving firewall policy anomalies," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 3, pp. 318–331, 2012.

[14] T. Abbes, A. Bouhoula, and M. Rusinowitch, "Detection of firewall configuration errors with updatable tree," *International Journal of Information Security*, vol. 15, no. 3, pp. 301–317, 2016.

[15] N. B. Neji and A. Bouhoula, "NAF conversion: an efficient solution for the range matching problem in packet filters," in *Proceedings of the 12th International Conference on High Performance Switching and Routing*, 2011, pp. 24–29.

Nguyễn Mạnh Hùng tốt nghiệp đại học về Công nghệ Thông tin năm 1998 tại Học viện Kỹ thuật Quân sự và bảo vệ luận án tiến sĩ năm 2004 tại Trung tâm Tính toán, Viện hàn lâm Khoa học Liên bang Nga. Hiện nay, ông đang công tác tại Phòng Sau đại học, Học viện Kỹ thuật Quân sự. Các hướng nghiên cứu tác giả đang triển khai là cấu trúc dữ liệu hiện đại, khai phá dữ liệu, xử lý song song và khớp ontology.

Vũ Duy Nhất tốt nghiệp đại học về Công nghệ Thông tin năm 2002 tại Học viện Kỹ thuật Quân sự. Hiện nay, ông đang công tác tại Cục Cơ yếu, Bộ tổng tham mưu. Lĩnh vực nghiên cứu quan tâm của ông là bảo mật công nghệ thông tin.